Class

1

CSC 141 Computer Science I

Introduction

Dr. Si Chen (schen@wcupa.edu)

Tue / Thu · 11:00 AM – 12:15 PM · Mitchell Hall 102
Fall 2026 · Week 1 · Tuesday, August 25, 2026
No prior programming experience required · wcu.ninja/csc141

PART 1

What is this course?

Programming, problem-solving, and why 2026 is a strange and exciting time to start

The one-sentence version

- CSC 141 is an **introduction to programming using Python** — building software from the ground up.
  - Turn a fuzzy human problem into precise, step-by-step instructions a computer can run.
- We cover: program design, variables & types, decisions, loops, functions, strings & sequences, and file I/O.
  - Counts as a **Science Distributive** general-education course.
- The real skill is not "Python syntax" — it is **computational thinking**: decomposition, precision, debugging.

By the numbers

5 programming assignments — 30%

5 short quizzes — 15%

2 midterms — 30%

Final exam — 20%

Attendance / participation — 5%

What you will be able to do by December

- Design algorithmic solutions using **decomposition, pseudocode, and stepwise refinement**.
- Write, run, and **debug** Python programs with variables, expressions, and I/O.
- Use Boolean logic, `if/elif/else`, and `while/for` loops to control program flow.
- Define and call **functions** with parameters and return values.
- Process **strings and sequences** with indexing, slicing, and iteration.
- Read and write **files**, and test/debug **systematically** — hypothesis, evidence, revise.

PART 2

The AI moment

Why learning to program changed more in the last 24 months than in the previous 20 years

The world you are walking into

75%

of new code at Google is AI-generated and approved by engineers (up from 50% in late 2025)

Pichai, Google Cloud Next, Apr 2026

~50%

of all committed code in AI-using organizations is now machine-written

Veracode 2026 GenAI Report

#1

Python is the most popular language on Earth — and the language of AI itself

TIOBE Index, 2026

You are learning the most in-demand language on Earth, at the exact moment machines started writing most of the code. That sounds like a threat. Today I want to convince you it is the opposite.

The honest picture: a hard job market for beginners

The scary headlines

- Entry-level SWE postings **dropped ~67%** (2023→2024).
- Recent CS grads: **~6.1% unemployment** — above the ~5.7% for all grads.
- Early-career workers (22–25) in AI-exposed jobs: **13% relative decline** since generative AI.
- CS enrollment fell **>8% at four-year schools** — first national drop in ~20 years.

The rest of the story

- Software-developer jobs still projected to **grow 15% (2024→2034)** — ~5× the average job.
- AI/ML roles face a **63% talent shortage**, 500k+ open roles globally.
- The jobs that vanished were **routine junior tasks** — not "understanding how software works."
- People who can **direct, verify, and debug** AI are the ones being hired.

> *When you ask the AI to do it for you, on the surface it might just seem it is writing the code. But under the hood, it is also doing the understanding of the task for you.*

— Murtaza Ali, University of Washington (Fortune, Aug 2026)

So why still learn to code by hand? Three reasons

AI is confidently wrong

Hallucinates names, off-by-one errors, logic that "looks right."

Only ~56% of AI code passes basic security checks.

You cannot catch a bug you do not understand.

You become the pilot

The new job is to specify, review, verify — not type.

A pilot who cannot fly manually cannot judge the autopilot.

Reading & debugging code is now core literacy.

Understanding compounds

Every concept here underlies everything AI builds.

Fundamentals let you learn the next tool in a week.

This is why we start from zero — on purpose.

How AI fits into CSC 141 — the ground rules

- **Allowed:** conceptual explanations, decoding error messages, extra practice problems, general debugging strategy.
- **Not allowed on graded assignments:** having AI generate substantial parts of the solution and submitting it as yours.
  - Quizzes and exams: no AI unless I explicitly say so.
- You must be able to **explain and modify every line** you submit — I may ask you to.
- Think of AI as a **tutor you question**, never a vending machine you copy from.

PART 3

How a computer actually works

Before we write code: what is a program, and what happens when it runs?

Hardware and software

Hardware

- **Hardware** is the physical machine:
- CPU — does the actual computing (arithmetic, logic).
- Main memory (RAM) — fast, temporary working storage.
- Storage, input (keyboard), output (screen).

Software

- **Software** is the instructions the hardware follows:
- A **program** is a precise, ordered list of instructions.
- The CPU does exactly what it is told — no more, no less.
- Your job in this course: write those instructions.

Programming languages: from machine code to Python

- **Machine language** — raw 1s and 0s the CPU executes directly. Fast, but unreadable to humans.
- **Assembly** — short mnemonics for machine instructions. Still very low-level.
- **High-level languages** (Python, Java, C) — readable, close to human ideas; one line does a lot.
  - We use **Python**: clean, readable, beginner-friendly — and the dominant language of data science and AI.
- A **translator** turns your high-level code into something the CPU can run.

How Python runs your code

Interpreted, line by line

- You write **source code** — plain text in a `.py` file.
- Python is **interpreted**: an interpreter reads and runs your code **line by line, top to bottom**.
- (Java/C are **compiled** first into a separate executable — Python skips that step, so it feels immediate.)
- This is why you can type a line and see the result right away.

Runs (almost) anywhere

- **Portability:** the same `.py` file runs on Windows, macOS, and Linux —
- ...anywhere Python is installed. Write once, run (almost) anywhere.
- You can even run Python **in a web browser** — try it: **wcu.ninja/csc141/python**.
- No machine details to worry about: focus on the logic.

PART 4

Let's actually run something

Your tools, your first program, and a hands-on exercise — bring a laptop

Our toolkit: Python 3 + Visual Studio Code

What we use

- **Python 3** — the language. Free; on lab machines and your own laptop.
- **VS Code** — a professional editor, the same one used across the industry.
- Install the **Python extension** (Microsoft) in VS Code.
- IDLE or another approved editor is accepted — we standardize on VS Code.

Why this choice

- Why VS Code? **Free**, cross-platform, industry standard.
- Built-in **terminal, debugger, IntelliSense** grow with you.
- The **same tool** from `print("hi")` to full projects.
- No setup today? Use the **browser playground** and follow along.

Setup in 4 steps — follow along

1 · Install Python 3

python.org (or lab machines).

Windows: check "Add to PATH".

Verify: `python --version`.

2 · Install VS Code

code.visualstudio.com.

Open it — your workshop.

3 · Python extension

Extensions → search "Python".

Install the Microsoft one.

Enables run + debug.

4 · Make a folder

Create `csc141` on Desktop.

File → Open Folder.

New File → `hello.py`.

Your first program — and what every piece means

```
print("Hello, CSC 141!")
print("My name is Ada.")

# This is a comment — Python ignores it
print("I am learning to program.")
```

- `print(...)` is a **function** that displays whatever is inside the parentheses.
- `"Hello, CSC 141!"` is a **string** — text, always in quotes.
- A line starting with `#` is a **comment** — notes for humans, ignored by Python.
- Lines run **top to bottom, in order** — exactly as written.
- Compare to Java: no `class`, no `main`, no `System.out.println` — Python is direct.

Run it: click ▶ (top-right) or press Ctrl+F5. Watch the terminal appear at the bottom.

Now you try — make it yours

HANDS-ON ⏱ 8–10 min

```
print("Hi, I am ___")

# say something about yourself
print("I like ___")

# now delete a " and run it —
# what does Python say?
```

- Open **wcu.ninja/csc141/python** (or VS Code) and write a short program that:
  - prints a **greeting** with your name,
  - prints **one fact** about you on its own line,
  - includes at least **one comment** (#) explaining what it does.
- **Then break it on purpose:** delete a quotation mark and run it. Read the error message.
- **Fix it.** That edit → run → read → fix loop is the whole job.

Recap — and where Class 2 picks up

- A program is **precise, ordered instructions**; the CPU follows them literally.
- Python is **interpreted**, runs line by line, and is **portable** — even in the browser.
- You wrote and ran real code, and met the **edit → run → read → fix** loop.
- **Thursday (Class 2):** how the computer *computes* — values, types, expressions, and `print()`/`input()` in depth.
  - An AI could have written today's lines instantly — but now *you* understand them. That is the point.

Before Thursday

- Get **Python 3 + VS Code + the Python extension** working on your own laptop (office hours / lab if stuck).
- Re-run today's program from scratch, without looking. Try the browser playground too.
- Read the **syllabus** on the course site & D2L — grading, late-work, and the AI policy.
- Course site: **wcu.ninja/csc141**. No graded work yet — Programming Assignment 1 comes in Week 2.

Sources & further reading

- Google 75% AI-generated code — Pichai, Google Cloud Next 2026 (Fast Company; DevOps.com)
- Veracode 2026 GenAI Code Security Report — 56% pass rate (veracode.com)
- TIOBE Index 2026 — Python #1 (TechRepublic; TIOBE)
- Stanford Digital Economy Lab — "Canaries in the Coal Mine," 13% early-career decline
- Entry-level SWE postings −67% (2023→2024) — Stanford Digital Economy Lab
- CS enrollment −8% at four-year schools — National Student Clearinghouse (Fortune, Aug 2026)
- Murtaza Ali quote — Fortune, "vibe coding college," Aug 3 2026
- BLS: software developer employment +15% 2024–2034 — bls.gov